



The PHP Company

Utiliser le Zend Framework avec Symfony

Xavier Gorse - ELAO

A partir du support de
Matthew Weier O'Phinney
Chef de Projet, Zend Framework

Xavier Gorse - xavier.gorse@elao.com

- Gérant de la société ELAO
- Expertise, formation et développement d'application Symfony
- Ancien président de L'AFUP
- Membre de PHPTV



Matthew Weier O'Phinney

- Développeur PHP depuis 2000
- Contributeur du Zend Framework depuis 2006
- Chef de projet depuis 2009



Zend Framework : quelques rappels

Le Zend Framework est ...

- une bibliothèque de composants ?
- un framework full-stack ?
- Réponse : les deux.

Core

Registry

Uri

Log

Cache

Mail

Config

Loader

Date

MVC

Controller

View

Layout

Application

Formats & Data Access

Db

Json

Search

Dom_Query

Ldap

Feed

Amf

Queue



Security

Auth

Acl

Filter

Validate

OpenId

Captcha

Web Services

XmlRpc

Soap

Rest

Http_Client

Amazon

InfoCard

Yahoo

Flickr

Twitter

Akismet

Delicious

...

Web Infrastructure

Session

Tag

Form

Navigation

ProgressBar

Dojo

Internationalization

Locale

Translate

Currency

Measure

Development

Tool

WildFire

Debug

Test

Comparatif Symfony / Zend Framework

Quelle importance ?

- Tous deux sont des frameworks éprouvés
- Chacun à sa manière soulage le développeur
- Ils peuvent être utilisés conjointement
(grâce aux composants Symfony, cela marche dans les deux sens !)

Pourquoi utiliser ZF avec Symfony?

Tirer profit des fonctionnalités du ZF

- Accès aux APIs distantes (web services, flux RSS, etc.)
- Support de l'indexation à l'aide de Lucene
- Generation de PDF
- Mise en file d'attente
- Cloud computing (stockage, bases de données, files de messages)

Traitements asynchrones

- Généralement : déporter certains traitements sur d'autres composants
 - ▶ Files d'attente
- Exemples:
 - ▶ Envoi de notifications de mails
 - ▶ Interaction avec des web services tierces
 - ▶ Opérations en base de données coûteuses ou complexes
- Cela peut nécessiter de lancer une console ou d'interagir avec une file de messages

Exposer des Web Services

- XML-RPC
- SOAP
- JSON-RPC
- AMF

Comment utiliser le Zend Framework avec Symfony?

Méthodes d'intégration

- Généralement, en installant un plugin ou en configurant le projet

```

class ProjectConfiguration
    extends sfProjectConfiguration
{
    static protected $zendAutoloader = false;

    static public function registerZend()
    {
        if (!self::$zendAutoloader) {
            set_include_path(implode(
                PATH_SEPARATOR, array(
                    sfConfig::get('sf_lib_dir') . '/vendor',
                    get_include_path(),
                )))
            require_once 'Zend/Loader/Autoloader.php';
            self::$zendAutoloader =
                Zend_Loader_Autoloader::getInstance();
        }
        return self::$zendAutoloader
    }
}

```

Allons un peu plus loin ...

```
class ProjectConfiguration
    extends sfProjectConfiguration
{
    // ...

    // Autoload PEAR classes, too...
    static public function registerPear()
    {
        self::registerZend()->setFallbackAutoloader();
    }
}
```

Utilisation des composants Zend :

```
class readerReadTask extends sfBaseTask
{
    // ...
    public function execute(
        $arguments = array(),
        $options = array()
    ) {
        // ...
        ProjectConfiguration::registerZend();
        $feed = Zend_Feed_Reader::import($feed);
        // ...
    }
}
```

Il ne reste plus qu'à utiliser les classes!

Quelques composants qui peuvent vous
être utiles ...

Les composants “service” (Zend_Service)

- **Zend_Gdata**
 - ▶ Contacts, calendriers, and YouTube!
- **Zend_Service_Amazon**
 - ▶ Recherche de livres, S3 & EC2, SQS (prochainement)
- **Zend_Service_Akismet**
 - ▶ Détection de spams
- **Et beaucoup, beaucoup d'autres**
 - ▶ <http://framework.zend.com/manual/fr/zend.service.html>

Les outils de syndication (Zend_Feed)

- **Zend_Feed_Reader** - support complet pour consommer des flux, dont RSS (1 and 2) & Atom
- **Zend_Feed_Writer** - le “miroir” de Zend_Feed_Reader, destiné à rédiger des flux
- **Zend_Feed_Pubsubhubbub** - interaction avec les hubs PuSH, facilite la publication de flux et la souscription

```
$feed = Zend_Feed_Reader::import($feedUri);
$metadata = array(
    'title'          => $feed->getTitle(),
    'description'    => $feed->getDescription(),
);
$entries = array();
foreach ($feed as $entry) {
    $entries[] = array(
        'title'       => $entry->getTitle(),
        'link'        => $entry->getLink(),
        'timestamp'   => $entry->getDateModified(),
    );
}
```

```
$feed = new Zend_Feed_Writer_Feed;  
$feed->setTitle($someTitle)  
    ->setLink($url)  
    ->setDescription($description);  
$entry = $feed->createEntry();  
$entry->setTitle($entryTitle)  
    ->setLink($entryLink)  
    ->setDateModified(time());  
$feed->addEntry($entry);  
echo $feed->export('atom');
```

L'indexation avec Lucene

- **Lucene** est un format binaire optimisé pour l'indexation de documents et la recherche selon des critères complexes
- **Zend_Search_Lucene** traite les index au format Lucene
 - ▶ Supporte la recherche et l'exploitation de données basée sur des index au format Lucene
 - ▶ Supporte la generation d'index au format Lucene

```
if (is_dir($indexDir)) {
    $index = Zend_Search_Lucene::open($indexDir);
} else {
    $index = Zend_Search_Lucene::create($indexDir);
}
$doc = new Zend_Search_Lucene_Document();
$doc->addField(Zend_Search_Lucene_Field::Keyword(
    'uri', $url));
$doc->addField(Zend_Search_Lucene_Field::UnIndexed(
    'timestamp', $ts));
$doc->addField(Zend_Search_Lucene_Field::Text(
    'synopsis', $synopsis));
$doc->addField(Zend_Search_Lucene_Field::Unstored(
    'content', $content));

$index->addDocument($doc);
```

```
$hits = $index->find('+hello -dolly');  
foreach ($hits as $hit) {  
    printf('<a href="%s">%s ... (created %s)</a>',  
        $hit->uri,  
        $hit->synopsis,  
        $hit->timestamp  
    );  
}
```

Traitement des fichiers PDF

- PDF = une specification ouverte publiée par Adobe
- **Zend_Pdf** permet de :
 - ▶ Manipuler et exploiter des fichiers PDF existants
 - ▶ Créer de nouveaux fichiers PDF

```
$pdf = Zend_Pdf::load($pdfFile);  
$page = $pdf->pages[0];  
$font = Zend_Pdf_Font::fontWithName(  
    Zend_Pdf_Font::FONT_HELVETICA  
);  
$page->setFont($font, 36)  
    ->setFillColor(  
        Zend_Pdf_Color_Html::color('#CCC'))  
    ->drawText('U R H2O-marked', 60, 500);  
$pdf->save($pdfFile);
```

Sortons du cadre : Le cas des objets métiers

Pour quelle raison ?

- Exécuter des tâches de traitement séparées et distinctes, sans recourir à l'artillerie lourde proposée par le framework
 - ▶ Lorsqu'il est nécessaire de privilégier la vitesse d'exécution
 - ▶ Lorsqu'il faut réduire au strict minimum les ressources utilisées
 - ▶ Pour effectuer des traitements qui ne sont pas spécifiques à l'application Web (commandes en mode console, opérations en base de données, dialogue avec des web services, tâches périodiques, etc.)

Autres raisons :

- Tester unitairement les objets métiers
 - ▶ Les tests sur la base de données ne sont pas des tests unitaires
 - ▶ Raison principale : rendre les batteries de tests moins complexes
- Réduire les dépendances
 - ▶ Peut réduire la quantité de ressources utilisées
 - ▶ Peut améliorer les performances (moins de bootstrapping et moins de ressources = code plus rapide)
- Utiliser les objets dans différents contextes

Rôle des objets métiers

- **Entities** - souvent des “*POPO*” (Plain Old PHP Objects)
- **Données agrégées ou Collections** - d'entities
- **Mappers** - mapping entre les entities et les données persistantes, et vice-versa (essentiellement le rôle des ORM). Retourne des collections d'entities ou des données agrégées.
- **Les objets de la couche Service** - API publique de vos objets métiers ; ils utilisent d'autres objets du domaine et fournissent la logique métier

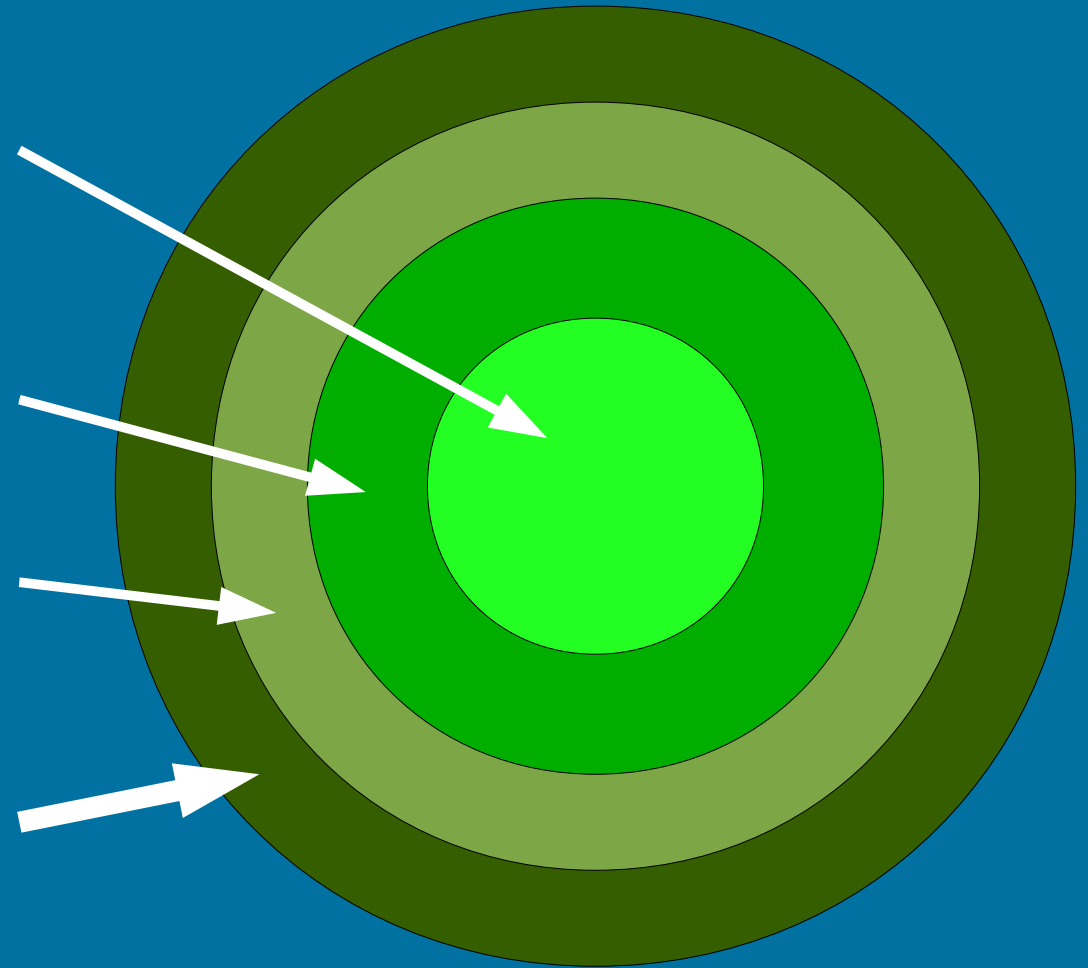
La couche “Service”

Data Access Objects
and Data store(s)

Data Mappers

Entities

Service
Layer



Quelques conseils

- Les objets de la couche Service ne devraient retourner que des entités et/ou des données agrégées
- Pensez à retourner des objets Zend_Paginator
 - ▶ Implémentant l'interface Countable
 - ▶ Implémentant l'interface Iterable
 - ▶ Les utilisateurs de la couche Service n'ont pas à savoir comment les données ont été retournées

Autres conseils

- Implémenter la logique métier dans la couche Service
 - ▶ Validation/Filtrage des données
 - ▶ Interactions entre les entités
 - ▶ Contrôles d'accès (ACLs)

Retour dans le cadre : Les traitements asynchrones

Les files d'attente (queues)

- Reporter à plus tard l'exécution d'un traitement
- Cas d'utilisation
 - ▶ Envoyer un email
 - ▶ Mettre à jour des index
 - ▶ Interagir avec des API tierces (Salesforce, SugarCRM, etc)

```
$queue = new Zend_Queue( 'MemcacheQ', array(  
    'name' => 'my-uber-queue',  
    'host' => 'queue.host.tld',  
));  
$queue->send( 'Some message' );
```

```
$messages = $queue->receive(5);  
echo count($queue), " messages registered\n";  
foreach ($messages as $message) {  
    // do something with message,  
    // typically $message->body  
    $queue->deleteMessage($message);  
}
```

Exposer des Web Services

Objectifs d'un web service bien conçu

- S'auto-documenter
- Fournir des réponses rapides
- Etre lié aux objets de la couche Service

Type de services fournis par ZF

- **AMF** (Active Message Format) - utilisé par Flex & Flash
- **JSON-RPC** - RPC utilisant JSON comme format de serialization ; clients en Dojo, YUI et ExtJS
- **XML-RPC** - protocole standard utilisé partout
- **SOAP** - service fourni avec l'auto-generation du WSDL et la serialization vers et à partir d'objets PHP

Procédure élémentaire

- Tous les serveurs se conforment à l'API SoapServer de PHP
 - ▶ *Instanciación*
 - ▶ *Ajout de classes et/ou de fonctions de callback*
 - ▶ *Traitement de la requête*

```
$server = new Zend_XmlRpc_Server();  
$server->setClass('My_Service_ApiClass', 'api');  
echo $server->handle();
```

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {  
    $server = new Zend_Soap_AutoDiscover();  
} else {  
    $server = new Zend_Soap_Server($thisScript);  
}  
$server->setClass('My_Service_ApiClass');  
$server->handle();
```

Les objets de la couche Service sont rois !

- Les méthodes publiques sont exposées par défaut
- Définissez votre API publique dans la couche Service, et réutilisez-la en fonction de vos besoins
 - ▶ Les contrôles d'accès, la validation, etc., sont déjà définies et encapsulées

Ressources Ajax

- Les requêtes AJAX doivent être rapides
 - ▶ Pour les requêtes de type RPC, Zend_Json_Server est votre ami
 - ▶ JSON-REST est de plus en plus populaire ; envisagez une solution RESTful MVC légère pour consommer vos objets Service, ou bien l'API REST de Symfony
- Une fois de plus, les objets de la couche Service sont vos amis

Récapitulons

-
- Le Zend Framework fournit un large éventail de fonctionnalités utilisables dans vos applications Symfony.
 - Pensez à bâtir un modèle métier riche et indépendant de l'application qui s'appuie sur des composants variés.
 - Déportez les traitements chaque fois que vous le pouvez, et placez vos points d'entrée Service en dehors de l'application Web pour de meilleures performances.

Ressources

- Le manuel de ZF :
<http://framework.zend.com/manual>
- Ces diapositives sur Slideshare :
<http://slideshare.net/weierophinney>
- Notez cette conférence :
<http://joind.in/1413>

Merci de votre attention !